データ共有型Webアプリケーションにおけるサーバ暗号化

AUTHOR(S):

川本, 淳平; 吉川, 正俊

CITATION:

川本, 淳平 ...[et al]. データ共有型Webアプリケーションにおけるサーバ暗号化. 情報処理学会シンポジウムシリーズ 2007, 2007(3): 3A-1

ISSUE DATE:

2007-11

URL:

http://hdl.handle.net/2433/147388

データ共有型 Web アプリケーションにおけるサーバ暗号化

川 本 淳 平† 吉 川 正 俊†

Web アプリケーションは，従来デスクトップ上で実装されていたアプリケーションを Web 上で利用可能にしただけでなく，ユーザ同士が簡単にデータ共有を行え，コラボレーションによって新たな知を生み出せる枠組みを提供している．しかし，こうした Web アプリケーションにおいては，サーバにプライバシ情報が収集されてしまうという問題が指摘されている．本研究では，この問題に対処するためにデータを暗号化して送信するサーバ暗号化を利用する．その上で，暗号化されたサーバ上のデータを，ユーザ間で共有するためのアクセス制御の導入や Web アプリケーションが提供するアカウントの機能を用いたデータクラスタリングによるアクセス速度の改善について提案する．

Server Encryption for Data Sharing Web Application

Junpei KAWAMOTO † and Masatoshi YOSHIKAWA†

Web applications provide not only services on the Web but also the place in which the users share data easily and create new wisdom by collaboration. At the same time application servers are collecting user data implicitly for render of service. Therefore, users' privacy data might be collected by servers. To overcome this problem, in this paper, we propose a system that encrypts sending data for server and decrypts received data using paring encycription. Also, our system provides the access control for sharing encrypted data. We also discuss performance enhancement of the system by data clustering using account service of servers.

1. はじめに

従来デスクトップアプリケーションとして提供されていたサービスを，Web を通しブラウザ上で実現する Web アプリケーションが急速に広まっている [1)2)]．こうした Web アプリケーションの特徴として次の 3 つを上げることができる．

(1) データが Web サーバ上で管理されるため，わざわざ持ち運ぶ必要がない．
(2) Web に接続できる端末があれば，いつでもどこからでもアプリケーションを利用できる．
(3) データが一箇所で管理されるため，他のユーザとの共有・連携が簡単に行える．

特に，3 番目に挙げたユーザ間でのデータ共有及び連携により，コラボレーションが活発に行われ，新たなコンテンツを生み出している．しかし同時に，Web アプリケーションを提供するアプリケーションサーバには次の問題も指摘されている．

- セキュリティに配慮したデータ管理を提供しない場合がある．
- ユーザデータを収集し利用することがある．

こうしたことから，Web アプリケーションの利用により，プライバシ情報が漏洩してしまうことが危惧されている．

Web アプリケーションは，その特徴により，人々が何時でも何所にいてもコラボレーションを行うことを可能とする．すなわち，ユビキタス・コラボレーションを実現するためには欠かせないサービスである．人々が安心して Web アプリケーションを利用し，ユビキタス・コラボレーション環境を実現するためには，アプリケーションサーバ側が先のプライバシ問題解決に協力し次の機能を提供することが不可欠である．

- ユーザデータは暗号化してサーバへ送信する．
- サーバでは暗号化されたまま保存し，サーバ自身によるデータの閲覧を不可能にする．
- 暗号化データに対してアプリケーションを実行するために必要な問合せをサポートする．

しかし，現在そのような機能を提供している Web アプリケーションは著者らの知る範囲ではほとんどない．そこで本研究では，現在提供されている Web アプリケーションにおいて，そのアプリケーションが提供する機能のみを利用し，サーバ側に特別な機能を仮定することなく上記プライバシ保護機能を実現する手

† 京都大学大学院情報学研究科社会情報学専攻
Department of Social Informatics, Graduate School of Informatics, Kyoto University

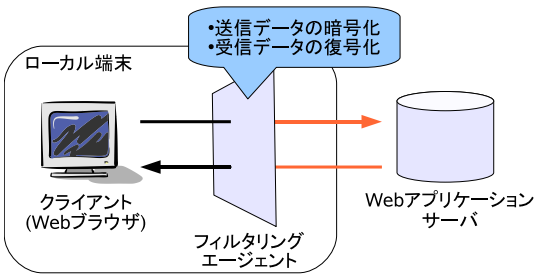

図 1 Web 通信フィルタリング・エージェントシステム

法について提案する．

プライバシ情報の漏洩を防ぐには，図 1 に示す様にデータを暗号化して送信するフィルタリング機能をローカル端末上で動作させることが有効である．外部のサービスを用いず，ローカル端末で暗号化することで，最も高いレベルでのプライバシ保護が行える．しかし，単純にデータを暗号化して送信した場合，次のような問題が発生してしまう．

- 他のユーザとのデータ共有が行えない．
- データの検索が行えない．

本論文では，ユーザとのデータ共有が行えないという問題を解消するために，ペアリング暗号とユーザのグループ化を用いたアクセス制御機構の構築方法を提案する．また，データの検索問題に対してもメタデータを付加する改善策を紹介する．

本論文の構成は，2 章で関連研究について，3 章では本論文で提案する暗号化データ共有のためのアクセス制御方式の概要について述べる．4 章では 3 章で紹介する手法に対し，ペアリング暗号とデータクラスタリングを用いた高速化について説明し，5 章で暗号化データに対する問合せの実現方法を紹介する．6 章では本提案手法を Web アプリケーションに対し導入することを考え，具体的なアルゴリズムについて説明する．最後に 7 章で実験を行い，8 章でまとめと今後の課題について述べる．

2. 関 連 研 究

2.1 組織構造を用いたアクセス制御

大田ら [3] は，データベースサーバに暗号化してデータを保存することで，ユーザのその情報への参照を制御する仕組みについて提案している．この研究では，データベースサーバを利用する組織の階層構造に注目して，各ユーザが管理すべき鍵の数やデータに付加されるアクセス制御情報の削減を行っている．

それに対し本論文では，Web アプリケーションを利用するユーザ間に組織的構造を仮定せず，平等な関係の上で実現を行っている．

2.2 暗号化データベースに対する索引付け

三浦ら [4] は，サーバの管理者にデータ内容が知られてしまう問題に対するため，クライアント側でデータを暗号化するサーバ暗号化について述べており，暗号化したデータに対する検索を行うための索引付けについて提案している．

それに対し本論文では，サーバ暗号化の技術の上で，どのようにしてデータ共有という Web アプリケーションの特徴を実現するかについて述べている点で異なっている．

3. 暗号化データ共有のためのアクセス制御

本研究では，前述の様に，プライバシ保護の仕組みとしてローカル端末上で動作する Web 通信フィルタリング・エージェントシステムを考える．このシステムを利用するユーザは，フィルタリングが行われていることを意識せずに透過的に Web アプリケーションにアクセスできなければならない．そのために，ユーザインターフェース上ではユーザのアクセスできないデータは表示されず，以下で説明する複数アカウント方式であっても一つのアカウント上で動作している様に見える必要がある．本章では，このフィルタリングシステムにおいて，暗号化されたデータを共有するためのアクセス制御の実現方法について述べる．

以降では，本手法を用いたデータ共有システムを利用するユーザ集合を U とし，ユーザ数を $n = |U|$ とする．また，ユーザ集合の中で実際にデータの共有を行うユーザの集合を $S_i \subseteq U$ $(i = 1, 2, \cdots)$ とする．従って，U の中にはシステムは利用するがデータの共有関係に無いユーザの組が含まれている可能性もある．

3.1 Web アプリケーションにおけるアカウント

Web アプリケーションでは，いくつかのデータの集まりをアカウントと言う概念を用いて管理している．例えば，1 人のユーザに関するデータを 1 つのアカウントとして管理するなどである．また，1 人のユーザが複数のアカウントを持つこともできる．アカウントにはユーザ識別子とパスワードによる認証が提供されており，Web アプリケーションを利用する上での大まかなデータ集合を構築することができる．

本研究では，このアカウントを本来 Web アプリケーションが想定している利用法では無く，本エージェントシステムがアクセス制御機能を提供するために用いる．従って，アカウントに設定されている識別子とパスワードはエージェントが管理しユーザには通知しない．そのため，ユーザはシステムを介さずアカウント

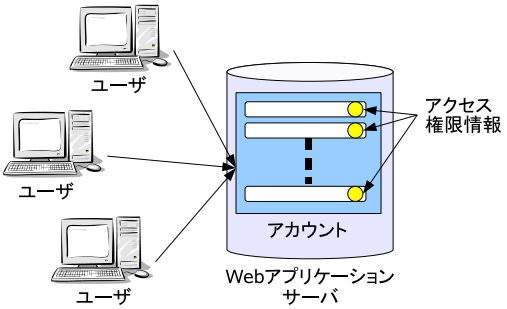

図 2 一つのアカウントを利用する手法

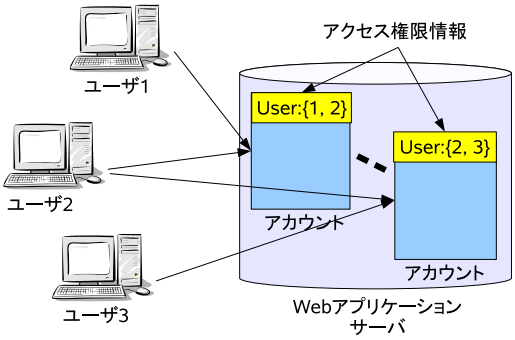

図 3 複数のアカウントを利用する手法

にアクセスすることはできず，本システムが想定しない操作をアカウントに行うことはできない．また，本システムを利用するためのユーザ認証機能はシステムが別に用意するものとする．

アカウントを用いたアクセス制御には 2 通りの手法が考えられる．

(1) 一つのアカウントを利用する手法.
(2) 複数のアカウントを利用する手法.

以降では，これらについて紹介し，それぞれの利点と問題点について説明する．

3.2 一つのアカウントを利用する手法

一つのアカウントを利用する手法では，図 2 に示す様に，すべてのユーザが一つのアカウントに対し読み書きを行う．一つのアカウントを複数のユーザで共有利用する場合，他人のデータに対する改変及び削除が問題となる．しかし，前述のようにアカウントに対する操作は本システムを通さずには行えない．従って，不正なアクセスはシステムが取り除き，こうした問題は発生しないと言える．また，アカウントに読み書きするためのパスワードは各ユーザのエージェントに前もって知らせておくものとする．

このモデルでアクセス制御を行う為には，新たなデータエントリ e が追加される時，次に示す手順で暗号化を行う．

(1) e を共通鍵 k_e を用いて暗号化し $Enc(e, k_e)$ を得る．
(2) 共通鍵 k_e をアクセス権限を与えるユーザの公開鍵 $pub_1, pub_2, \cdots, pub_m$ で個々に暗号化し付加する．

以上の手順により得られた，

$$Enc(e, k_e) \oplus \bigoplus_{j=1}^{m} Enc(k_e, pub_j)$$

をアカウントへ追加する．ここで，演算子 $\oplus$ は文字列の連結を表すものとする．データエントリを利用する場合，エージェントはユーザの秘密鍵を用い $\bigoplus_{j=1}^{m} Enc(k_e, pub_j)$ に対して順に復号を試みる．復号化でき共通鍵 k_e を得られればアクセス権限が与えられていると分かり，データエントリ e を手に入れることができる．

この手法では，データエントリ毎に細かい権限設定が可能であるが，

$$O(\text{データエントリ数}) \times O(\text{ユーザ数 } n)$$

に相当する膨大なデータがアカウントに送信される．また，データエントリを取得する場合，アクセス権限があるか否かは復号を試みるまで分からない．そのため，多くの不要のデータエントリに対し復号を試みる必要がある．一般に復号にはコストがかかるため，アクセス権限の無い多くのデータに対し復号を試みることは好ましくない．

3.3 複数のアカウントを利用する手法

複数のアカウントを利用する手法では，図 3 に示す様に，いくつかのアカウントを利用しアカウント毎に読み書きできるユーザ集合 S_i を設定する．この場合も先ほどと同様に，一つのアカウントを複数人で共有するが，不正操作はシステムが遮断するものとする．また同様に，アカウントに読み書きするためのパスワードは各ユーザのエージェントに前もって知らせておくものとする．このモデルでアクセス制御を行う方法は次のようになる．

3.3.1 準　　備

先ず，次の手順でアカウント毎に共通鍵暗号の鍵を設定する．

(1) 各アカウント A_i に共通鍵 k_{A_i} を設定する．
(2) 各アカウント A_i に対し，共通鍵 k_{A_i} をアクセス権限を与えるユーザの公開鍵で個々に暗号化したものをアカウントへ追加する．

例えば，$user_1, user_2, \cdots, user_m$ にアカウント A_i へのアクセス権限を与えるとし，ユーザの公開鍵をそれぞれ $pub_1, pub_2, \cdots, pub_m$ とすると，

$\bigoplus_{j=1}^{m} Enc(k_{A_i}, pub_j)$

というデータを作成し，アカウントへ追加する．

3.3.2 データエントリの追加

$user_1$ がアカウント A_i に新たなデータエントリ e を追加する場合，次の手順で行う．

(1) アカウント A_i からアクセス権限情報

$\bigoplus_{j=1}^{m} Enc(k_{A_i}, pub_j)$

を取得し復号を試みる．$user_1$ がアクセスが許可されている場合 k_{A_i} を取得できる．

(2) 追加するデータ e を k_{A_i} で暗号化し $Enc(e, k_{A_i})$ を A_i に追加する．

3.3.3 データエントリの取得

データエントリの取得も追加と同様に，先ずアカウントの共通鍵を得，それを用いて復号化する．逆にアクセスが許可されていない場合，アカウント用の鍵が手に入らないため，不正にアクセスされることは無い．

3.3.4 データ数の評価

複数のアカウントを利用する場合，サーバへ登録すべきアクセス権限情報は，

O(アカウント数) $\times$ O(ユーザ数 n)

となる．一般的に，

(アカウント数) $\ll$ (データエントリ数)

であるとすると，大幅にアクセス権限情報を減らすことができる．しかし，アクセス権限をアカウント単位でしか設定できないため，細かい設定が行い難い．例えば，データエントリが e_1, e_2, e_3 の 3 つあり，ユーザが $user_1$, $user_2$ の 2 人であるとする．この時，以下の様に権限を設定することを考える．

e_1: $user_1$ のみアクセス可能
e_2: $user_1, user_2$ 共にアクセス可能
e_3: $user_2$ のみアクセス可能

この時の解決方法は次の 2 つである．1 つ目は，アカウント A, B を作成し，

A: $user_1$ のみアクセス可能
B: $user_2$ のみアクセス可能

とする．そして，A に $\{e_1, e_2\}$ を登録し，B に $\{e_2, e_3\}$ を登録する．この場合，e_2 が複数のアカウントに登録され冗長となってしまう．また，同一のデータが複数個所に記録されるため，更新不整合が発生する可能性がある．そのため，更新不整合が起こらないように同期を取る必要があるが，同期にはコストがかかってしまい効率的ではない．2 つ目の方法は，アカウントを A, B, C の 3 つ作成し，

A: $user_1$ のみアクセス可能
B: $user_2$ のみアクセス可能
C: $user_1, user_2$ 共にアクセス可能

と設定する．そして A に $\{e_1\}$，B に $\{e_3\}$ を C に $\{e_2\}$ を登録する．この場合，アクセス権限設定の組み合わせに応じてアカウントの数が必要となり，最悪 $O(2^n)$ のアカウント数が必要となる．アカウントの数が増え，

(アカウント数) $\ll$ (データエントリ数)

が成り立たなくなると，一つのアカウントを用いた手法と同等のアクセス権限情報が必要になる．

次に，アクセス権限の有無を調べるための復号回数であるが，複数アカウント方式では，アカウント毎にアクセス権限の有無を調べるだけで良く，個々のデータエントリに対し復号を試みる必要はない．従ってアカウント数が少ない状況では，一つのアカウントを利用する手法に比べ復号を試みる回数は少なく効率的である．

4. ペアリング暗号とユーザクラスタリング

本章では，前章で紹介した 2 種類のアクセス制御機構の問題点について，ペアリング暗号とユーザのクラスタリングを用いた解決策について説明する．

4.1 ペアリング暗号を用いた権限情報の固定長化

一つのアカウントを利用する手法の問題点の一つは，アクセス権限情報に相当する共通鍵を暗号化したデータを数多く保管しなければならないことである．この問題の解決策として，ペアリング暗号を利用し権限情報の固定長化を行う．

ペアリング暗号とは，近年盛んに研究が行われている新しい公開鍵暗号のことである [5]．ペアリングとは，ある種の楕円関数上で定義される 2 入力 1 出力の関数 f のことで，次のように定められる．

$f : G \times G \longrightarrow G_T$ (G : 加法群, G_T : 乗法群)

この時，ペアリング関数 f は，以下で表される双線形性を持つ．

$$f(P^a, Q) = f(P, Q^a) = f(P, Q)^a$$
$$f(P_1 * P_2, Q) = f(P_1, Q) \cdot f(P_2, Q)$$

但し，$P, P_1, P_2, Q \in G$, $a \in Z$ とする．

このペアリング暗号を用いることで，アクセス権限を持つユーザの人数にかかわらず，固定長でデータエントリの共通鍵を暗号化することができる [6]．以下にユーザ数が n の場合について，その手順を簡単に記す．

4.1.1 準　　備

先ず，$P \in G$ なる P と，ランダムな整数 α, γ を生成する．次に，

$P_i = \alpha^i P$ $(1 \leq i \leq 2n)$
$Q = \gamma P$

により，$P_1, \cdots P_{2n}$ 及び Q を求める．この時，ユー

ザ i の秘密鍵は $D_i = \gamma P_i$ となる．また，全ユーザに P,$P_1 \cdots P_n$, $P_{n+2} \cdots P_{2n}$, Q は公開しておく．

4.1.2 暗　号　化

新たにデータエントリ e が追加される場合，乱数 τ を生成し，

$$K = f(P_{n+1}, P)^{\tau}$$

で計算される鍵 K を用いてデータエントリ e を暗号化し $Enc(e, K)$ を得る．そして，アクセス権限付加の為に，次式で C_0, C_1 を計算する．

$$C_0 = \tau P, \quad C_1 = \tau(Q + \sum_{j \in S} P_{n+1-j})$$

ここで，S はアクセス権限を付加するユーザ番号の集合である．これらを用いて，最終的にサーバに記録されるデータは，$Enc(e, K)$ と (C_0, C_1) となる．また，アクセス権限情報として付加される (C_0, C_1) のビット長はユーザ数 n に依らず一定となっている．

4.1.3 復　号　化

ユーザ i がデータエントリを復号化する場合，(C_0, C_1) を用いて，

$$K = \frac{f(P_i, C_1)}{f(D_i + \sum_{j \in S, j \neq i} P_{n+1-j+i}, C_0)}$$

より共通鍵 K を求め $Enc(e, K)$ を復号化し e を得ることができる．

4.2 クラスタリングによるアクセス数の減少

4.2.1 クラスタリングの導入

ペアリング暗号を用いることで，データサイズを増やさずにアクセス権限情報を付加することが可能となった．しかし，ユーザがあるデータエントリに対しアクセス権限を持っているか否かは復号を試みるまで分からず，復号コストの問題は残ったままである．そこで，ペアリング暗号方式と複数アカウント方式を組み合わせ，ユーザのクラスタリングを行うことでアクセス権限の有無を調べるデータエントリ数を減少させる．

複数アカウント方式では，アカウント毎にユーザのアクセス権限を設定していた．以降では，ユーザがあるアカウントに対してアクセス権限を持っていることを，アカウントに所属していると言う．この時，各ユーザが閲覧可能なデータエントリは所属しているアカウントにのみ配置される．ペアリングと複数アカウントを利用した方式では，システムは先ずユーザがどのアカウントに所属しているか調べる．データの追加する行う場合は，所属しているアカウントに，上述のペアリングを用いた手法でアクセス権限を付加したデータを追加する．従って，エージェントは所属しているアカウントのデータに対してのみ復号を試みれば良く，アクセス権限を調べるデータエントリ数を減らすことができる．

また，ユーザは所属しているアカウント上すべてのデータエントリにアクセス権限を持つわけではない．なぜなら各データエントリには，ペアリング暗号によるアクセス権限が付加されているためである．従って複数アカウント方式の問題点であった，細かいアクセス権限設定を行うと必要なアカウント数が膨大になるという問題は起こらない．

以下では，ユーザが所属するアカウントがどのように決定されるかについて説明する．

4.2.2 準　　備

先ず始め，ユーザ i はアカウント A_i に所属しているものとし，初期状態で所属しているアカウント A_i をユーザ i のホームアカウントと呼ぶことにする．

4.2.3 共有関係の追加

次に，ユーザ集合 $S \subset U$ で共有するデータが追加されたとする．このとき以下の手順で，共有データを保存するアカウントの決定と所属情報の更新を行う．

(1) S の各ユーザが所属しているアカウント集合 Σ を求める．

(2) アカウント $A \in \Sigma$ について，

$$\overline{N}_A = |\{u |\ u \in S \ and \ u \notin A\}|$$

を求める．

(3) $\overline{N}_A$ が最小となるアカウント A を求める．

(4) 対象アカウントが複数ある場合，A に属するユーザが最も少ないものを選ぶ．

(5) A に所属していないユーザを A に所属させる．

$\overline{N}_A$ は，新しいデータをアカウント A に登録した場合に，所属アカウントが増えるユーザ数を表している．ユーザが所属するアカウントの数が増えれば，アクセス権限の有無を調べるデータエントリ数が増え，望ましくない．従って，$\overline{N}_A$ が最小となるアカウントにデータを送信する．また，$\overline{N}_A$ が最小となるアカウントが複数ある場合，所属ユーザ数が最も少ないアカウントを選んでいる．これは，所属ユーザが多いアカウントには，格納されるデータ数も多いと言え，アクセス権限を調べるデータ数が増えてしまうからである．

5. 暗号化データにおける問合せ

Web アプリケーションを利用するためには，サーバ上のデータエントリに対し問合せを行うことが必要となる．例えば，キーワードにマッチするデータエントリを取得することや，範囲指定された数値データを取得すること等は頻繁に行われる問合せと言える．しかし，データエントリが暗号化されている場合，そのままでは問合せ処理は行えない．

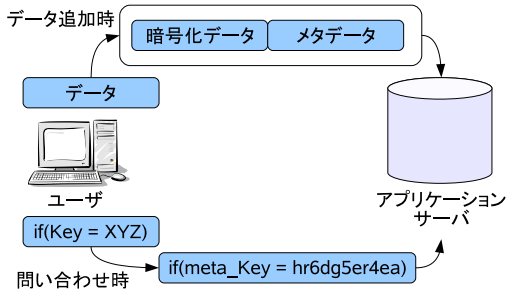

図 4 メタデータの利用

本節では，問合せに必要な判定を等号判定と不等号判定の 2 つに分け，それぞれを行うために暗号化データエントリに対してメタデータを付加する方法を紹介する．メタデータの利用は，図 4 に示す様に，データを追加する場合にメタデータを生成し付加する．問合せを行う場合，キーワードをメタデータに置き換えてから問合せを行う．

5.1 等号判定のためのメタデータ

キーワード一致検索や等しい数値データの検索などに用いる等号判定のためには，対象のキーワードや数値のハッシュを計算してメタデータとして付加する．ハッシュを用いると，ごく希ではあるが，ハッシュ値は一致するが求めるデータとは異なるものが得られる場合がある．この場合は実際に復号化して判断することになる．

5.2 不等号判定のためのメタデータ

ハッシュデータでは不等号判定を行うことはできない．そこで，三浦ら [4] により紹介されている手法を利用する．

今，対象の数値データが $0 \sim n$ の値をとる整数であるとする．これを範囲指定の粒度などを参考に k 個の連続な領域に分割する．ここでは簡単のため，

$$\{0 \sim \frac{n}{k}\}, \{\frac{n}{k} \sim 2\frac{n}{k}\}, \cdots, \{(k-1)\frac{n}{k} \sim n\}$$

に分割するが，各領域の大きさは等しくなくても良い．この各領域に対し，表 1 の様にランダムな数値を割り当てる．割り当てられたランダムな数値をメタデータとして用いる．例えば，$n/k \sim 2 \cdot n/k$ の間にあるデータに対しては，187651321 をメタデータとして埋め込む．

このメタデータを用いると，大まかな範囲判定を行うことができる．ハッシュ同様，問合せの結果得られるデータは，求めるデータと異なる物が含まれる可能性がある．こちらの場合も同様に，復号化を行い求めるデータを取得する．

領域	割り当てられた数値
$0 \sim n/k$	465138798
$n/k \sim 2 \cdot n/k$	187651321
⋮	⋮
$(k-1) \cdot n/k \sim n$	613579895

表 1 分割領域にランダムな数値を割り当てる

アカウント	開始日時	終了日時	イベント内容
1	200710151500	200710151600	情報システム
1	200710151630	200710151800	データベース
⋮	⋮	⋮	⋮

表 2 Web カレンダーが利用するデータ

6. Web アプリケーションへの適用

本章では，本研究で提案する手法を実際のアプリケーションへ適用する方法について述べる．ここでは，Web アプリケーションのモデルとして，バックエンドに関係データベースを利用しているものを考える．また，サンプルアプリケーションとして Web カレンダーを取り上げ，アプリケーションが利用するデータが表 2 のようになっているものとする．表中データの種類は次の通りとする．

アカウント ID: 整数
開始日時: 日付
終了日時: 日付
イベント内容: 文字列

Web カレンダーは，Web アプリケーションとして代表的であり，利用データの種類が数値，日付，文字列の 3 つを用いている．他のアプリケーションが利用するデータの種類も，大きく分ければ数値と日付と文字列の 3 種類である．従って，提案手法の適用は同じように行えると言える．

6.1 準　　備

先ず，ペアリング暗号を利用するための準備を行う．上述の計算式により公開情報を計算し，すべてのユーザ $u \in U$ に，$P, P_1, \cdots, P_n, P_{n+2}, \cdots, P_{2n}, Q$ を通知する．また，各ユーザ i に秘密鍵 D_i を通知する．次に，アカウント A_i $(1 \leq i \leq n)$ を取得し，パスワードを各ユーザのエージェントに通知する．以降では，A_i 番目のアカウント ID を id_i と書くことにする．

6.2 アカウント所属情報

ユーザはホームアカウント以外にも所属アカウントを持つ．そのため，各ユーザがどのアカウントに所属しているかの情報を登録する必要がある．これは，各アカウントに特殊なレコードを追加することで行う．例えば，ユーザ i がアカウント Aj にも所属し，シス

テムの利用開始日の前日を $preDay$ とすると，

- アカウント ID $= id_i$
- 開始日時 $= preDay$
- 終了日時 $= preDay$
- イベント内容 $= id_i, id_j$

をユーザ i のホームアカウントに保存する．開始日時と終了日時に設定する値は何でも良いが，他のデータと重複しないものが望ましい．他のユーザがユーザ i の所属しているアカウントを調べる場合，ユーザ i のホームアカウントから上記データを取得すれば良い．

6.3 データの追加

ユーザ i が，新たなデータエントリ e_{new} を追加する場合，システムがどのようなデータに変換するかについて説明する．ここでは，追加するデータ e_{new} が次の様であるとする．

- 開始日時 $= begin$
- 終了日時 $= end$
- イベント内容 $= contents$

また，e_{new} へのアクセス権限を与えるユーザ集合を $S \subseteq U$ とする．

6.3.1 ペアリング暗号化

先ず乱数 τ を生成し，公開情報 P，$P_1, \cdots, P_n$，$P_{n+2}, \cdots, P_{2n}$，Q より以下を計算する．

$$C_0 = \tau P,\ C_1 = \tau(Q + \sum_{j \in S} P_{n+1-j})$$

これらがアクセス権限情報となる．次に，本来のデータを暗号化する鍵 K を生成する．しかし，P_{n+1} は公開されていないため，以下の式により計算する．

$$K = f(P_{n+1}, P)^{\tau} = f(P_1, P_n)^{\tau}$$

また，追加するデータ e_{new} をテキストデータに変換したものを $Txt(e_{new})$ とすると，先ほどの共通鍵 K を用いて暗号化し，

$$Enc(Txt(e_{new}), K)$$

を求める．なお，e_{new} のテキストデータ化は，

開始日時 $= begin$
終了日時 $= end$
イベント内容 $= contents$
共有ユーザリスト $= user_i, user_j, \cdots, user_k$

と，各行に「$key = value$」形式で記述し，最後に共有するユーザ $u \in S$ の識別子を列挙するものとする．

6.3.2 メタデータの生成

サンプルアプリケーションでは，Web カレンダーの機能のうちイベントの開始日時での範囲指定問合せ，特に週単位での問合せと一日単位での問合せのみをサポートする．従って，付加するメタデータは上述の不等号判定のためのメタデータとする．

先ず，表 3 のように開始日時データを間隔 7 日で領域分割し各領域を表すランダム数値を求める．各領域のうち $begin$ が含まれる領域に割り当てられた数値を $meta_7$ とする．同様にして，間隔を 1 日で領域分割しメタデータ $meta_1$ を求める．

日時範囲	割り当てる数値
⋮	⋮
20070923 ～ 20070929	18674131
20070930 ～ 20071006	94134862
⋮	⋮

表 3 7 日間隔で領域分割しランダム整数を割り当てる

6.3.3 サーバへの送信

4 章で紹介したアルゴリズムにより求めた，送信先アカウントを id_l とすると最終的にサーバへ送信されるデータは次のようになる．

- アカウント ID $= id_i$
- 開始日時 $= meta_7$
- 終了日時 $= meta_1$
- イベント内容 $= Enc(Txt(e_{new}), K) \oplus (C_0, C_1)$

6.4 データエントリの取得

ユーザ i がデータを取得する場合，先ずホームアカウント A_i を調べ所属しているアカウント集合 Σ を得る．後は，$\forall A \in \Sigma$ に対し問合せを行う．問合せの結果，$Enc(Txt(e), K)$ 及び (C_0, C_1) を取得する．e に対しアクセス権限が与えられている場合，秘密鍵 D_i を用いて共通鍵 K を得ることができる．更に，この鍵を用いて $Txt(e)$ 及び e を得ることができる．

6.5 データエントリの更新

データエントリを更新する場合，更新後もアクセス権限を持つユーザ集合が同じであるなら，元々使用されていた共通鍵 K を用いて暗号化しサーバへ送信すれば良い．更新によってアクセス権限を持つユーザ集合に変更が生じる場合，先ず対象のデータエントリを削除する．その後，新しくアクセス権限を付加するユーザ集合を S_{new} として，上記データの追加手順を行う．

7. 実　　験

本論文で提案するユーザクラスタリングによって，アクセス権限の有無を調べる必要のあるデータ数が，どれだけ少なくなるか，アクセス権限適合率という指標を用いてシミュレーション実験を行った．なおアクセス権限適合率 r とは以下の式で定めるものとする．

$$r = \frac{(\text{アクセス権限を持っていたデータ数})}{(\text{アクセス権限の有無を調べる総データ数})}$$

実験では，どのユーザ間でデータの共有が行われるかをグラフで表現した．すなわち，図 5 に示す様に，

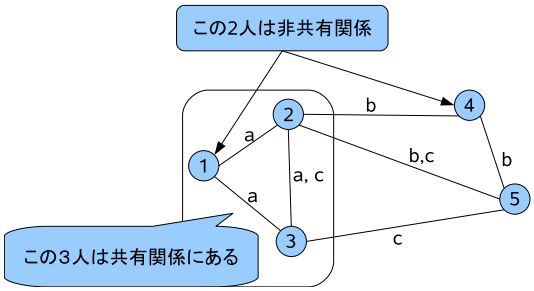

図 5 データ共有関係グラフ

一人のユーザを一つの頂点に割り当て，同一ラベルの枝が張られたユーザ間にデータの共有関係があるとする．図 5 では，ユーザ $\{1,2,3\}$ が共通の枝 a で結ばれており共有関係にある．同様に，ユーザ $\{2,4,5\}$, $\{2,3,5\}$ がそれぞれ枝 b，c で結ばれており共有関係にある．このデータ共有関係グラフを現実のデータ共有関係を反映させるために BA モデル [7] を基に作成する．データ共有関係グラフ生成のアルゴリズムは次の通りである．

(1) m 個の頂点 $n_1, n_2, \cdots, n_m$ からなり枝の無いグラフ G_0 をスタートとする．

(2) G_0 において，n_1, n_2, n_3 に対しお互いにラベル l_1 の枝を張ったものを G_3 とする．

(3) G_k において，n_{k+1} 以外の頂点の中から 2 つ選び a, b とする．但し，頂点が選ばれる確率はその頂点の次数に比例するものとする．

(4) G_k において，3 頂点 n_{k+1}, a, b をお互いにラベル l_{k+1} の枝で結んだものを G_{k+1} とする．

(5) (3)〜(4) 繰り返し G_m を作成する．

このようにしてユーザ数が 100 人，1000 人，10000 人の各場合について，共有関係グラフを作成し各ユーザに対して 1〜100 個の非共有データと各共有グループごとに 1〜50 個の共有データを格納させるシュミレーションをそれぞれ 3 回ずつ行った．その結果として，横軸に各ユーザが所属するアカウント数，縦軸にはアクセス権限適合率を取る平面に各ユーザをプロットしたものが図 6，7，8 である．また，アクセス権限の適合率は表 4 の様になった．

図 6，7，8 から分かるように，どの場合でも所属アカウント数が少ないユーザが最も多い．所属アカウントの少ないユーザの中には，アクセス権限適合率が高いユーザと低いユーザが共に含まれているが，相対的にアクセス権限適合率の高いユーザが多いと言える．表 4 において，適合率の平均値を高めているのはこれらのユーザであると言える．また，所属アカウント数が多いユーザもわずかながら存在する．これらのユーザのアクセス権限適合率は 0.5 強であり，多くのアカウントを参照しなければならない代わりに，適合率が極端に低くなることはないと言える．

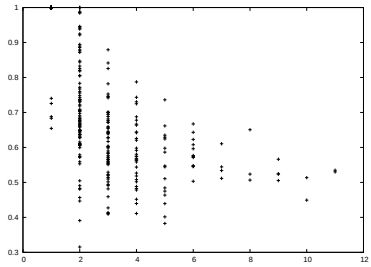

図 6 所属アカウント数とアクセス権限適合率の相関 (100 ユーザ)

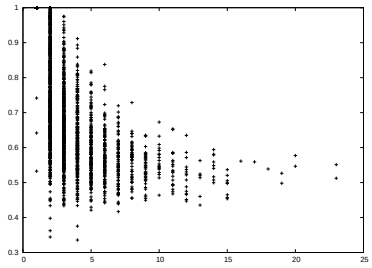

図 7 所属アカウント数とアクセス権限適合率の相関 (1000 ユーザ)

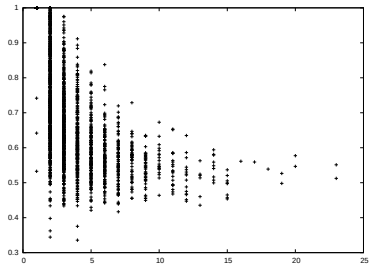

図 8 所属アカウント数とアクセス権限適合率の相関 (10000 ユーザ)

実験結果より，今後の課題として改善を行う必要があるのは，所属アカウント数は少ないがアクセス権限適合率も低いユーザと，所属アカウント数が多いユーザであると言える．後者に関しては，アクセス権限適合率は一定の水準を保っているが，参照するアカウントが増えれば，その分通信コストが掛ると言えるからである．

8. まとめと今後の課題

本論文では，Web アプリケーションを利用する上で，サーバによりユーザのプライバシデータが収集されてしまう可能性があるという問題を解決するために，サーバへ送信するデータを暗号化し，ペアリング暗号

ユーザ数	実験番号	平均値	最小値
100	1	0.7277	0.3251
	2	0.7412	0.3145
	3	0.7193	0.2424
1000	1	0.7762	0.2703
	2	0.7680	0.3333
	3	0.7727	0.2513
10000	1	0.7727	0.3980
	2	0.7756	0.3735
	3	0.7774	0.3957

表 4 アクセス権限の適合率

とユーザクラスタリングを用いた暗号化データに対するアクセス制御手法について論じた．暗号化データを扱う際の別の問題点として，暗号化データに対する問合せも重要である．本論文ではメタデータを用い，等号と不等号を判定する手法を紹介したが，複雑な問合せへの対応や計算コストの改善については今後の課題である．

また，本論文で提案する手法を Web カレンダーに対して適用したが，本提案手法が効果的な Web アプリケーションと，あまり効果的でない Web アプリケーションがあると言える．Web カレンダーでは，一度登録されたデータが変更されることは少なく，アクセス権限情報が更新されることも少ない．このようなアプリケーションにおいては，一度取得したデータをキャッシュすることができる．キャッシュを用いることで，暗号の復号コストを回避することができるため，本提案手法が効果的に利用できるアプリケーションと言える．逆に，頻繁にデータやアクセス権限が更新されるアプリケーションにおいては，キャッシュを用いることができないだけでなく，更新のたびに復号コストが生じる．そのため，本提案手法の適用が，あまり効果的でないアプリケーションであると言える．このようなアプリケーションの例としては，チャットなどを利用しリアルタイム性が重要となるグループウェアが考えられる．

今後の課題としては，所属アカウント数は少ないがアクセス権限適合率も低いユーザ及び所属アカウント数が多いユーザのコストを改善するために，一度格納されたデータの保存先を変更するなど，動的な再配置を行うことを考えている．

参 考 文 献

1) Google カレンダー:
`http://www.google.com/calendar/`
2) Remember The Milk:
`http://www.rememberthemilk.com/`
3) 大田 幸由, 清水 明宏, “暗号を用いた共有情報制御方式の検討”, 電子情報通信学会技術研究報告. OFS, オフィスシステム Vol.93, No.435(19940121) pp. 7-12, 1994.
4) 三浦 志保, 渡辺 知恵美, “管理者に対しても機密を保持できる暗号化データベースの索引構成法”, 電子情報通信学会データ工学ワークショップ (DEWS2007), E7-8, 2007
5) 岡本 栄司, 岡本 健, 金山 直樹, “ペアリングに関する最近の研究動向”, IEICE Fundamentals Review, Vol. 1, No. 1, pp. 51-60, July 2007.
6) D. Boneh, CGentry and B. Waters, “Collusion resistant broadcast encryption with short ciphertexts and private keys,” CRYPTO 2005, Lect. Notes Comput. Sci., vol.3621, pp.258-275, 2005.
7) A.-L. Barabási and R. Albert, “Emergence of scaling in random networks,” SCIENCE, vol. 286, pp. 509-512, Oct. 1999.